by
Jeff Prosise

Tutor

UNDOCUMENTED DOS FUNCTIONS

From time to time, I read about undocumented DOS functions in *PC Magazine*. Can you explain some of these functions and document their uses and calling conventions?

Stephen B. Coffman
Anaheim Hills, California

■ **UNDOCUMENTED DOS FUNCTIONS: These functions aren't described in any DOS manual, but they provide an interesting view of how DOS works.**

You bet. Figure 1 shows a partial list of undocumented DOS functions (marked *reserved* in most DOS technical references). There are more, but these are some of the most interesting—and useful. Whether you're a programmer or not, you'll find that these functions provide a working education on the internals of DOS.

One of the most useful undocumented DOS functions is function 60h, often called TRUENAME. It accepts an ASCIIZ filename (an ASCII string delimited with a binary zero) and returns an unambiguous, fully qualified filename that includes drive and path designators—even across networks. TRUENAME sees through the aliases set up by SUBST and JOIN. And in DOS 4.*x* it is actually an undocumented command that you can type at the command line. So, if you've used SUBST to make F: a synonym for the directory C:\DATA\FOO, then the command TRUENAME F:BAR will return C:\DATA\FOO\BAR. This is a service that many application programs have had to provide for themselves in the past. The segment:offset address of the original filename is passed in DS:SI; the address of the buffer that will receive the fully qualified version of the filename in ES:DI.

Functions 1Fh and 32h return the addresses of drive parameter blocks—tables DOS sets up that contain information about the drives installed in the system—above and beyond what's contained in the block device drivers associated with the drives. The format of the table (also shown in Figure 1) has changed with each major release of DOS, primarily to reflect changes in the operating system itself. In DOS 4.0, for example, the location that stores the length of each file allocation table in sectors (offset 0Fh) was changed from a byte to a word, in keeping with DOS 4.0's support for large partitions.

Function 37h changes the character DOS recognizes as the switch character, for example, the / in CHKDSK /F. The default character is a forward slash, but you can change it to anything you want by passing the ASCII code of the new switch character as an argument to function 37h in register DL. Or you can call it with AL set to 0 to obtain the current switch character.

You can create a short utility that changes the switch character from a forward slash to a hyphen (as in Unix) with this DEBUG script:

```
A    0100
MOV  AX,3701
MOV  DL,2D
INT  21
RET

N SWITCH.COM
RCX
08
W
Q
```

Note that this will work only with DOS commands; application programs may or may not respond to the switch. SWITCH.COM will work with DOS commands, but may or may not affect the way application programs interpret text on the command line. Most applications don't even bother to use function 37h to find out what the current switch character is; instead, they hard-code the switch characters.

RX FOR TSRS

TSR programmers find functions 34h, 50h, 51h, and 5Dh particularly useful. Function 34h returns the segment:offset address of the InDOS flag, a 1-byte flag that DOS increments when an interrupt 21h call is made and decrements when the call is completed. By checking InDOS before placing a call to a DOS function itself, a TSR can avoid the reentrancy problems that crop up when function calls are nested.

InDOS isn't the only internal flag TSRs need to access. They usually check a second flag, the *critical error flag*, before calling DOS functions. DOS sets this flag to a nonzero state when a critical error occurs, indicating to all concerned that the operating system is in an unstable state and function calls should be restricted. Function 5Dh, subfunction 06h returns the address of the critical error flag in Versions 3.*x* and 4.*x*. In earlier versions, finding the flag requires a difficult search through memory for a particular code sequence. The method is described in the *MS-DOS Encyclopedia*.

Although DOS is a single-tasking operating system, it maintains a record of the PSP address of the process that is currently in the foreground. When a TSR temporarily grabs control of the CPU, it should activate its own PSP before performing any file I/O, and then reactivate the formerly active PSP before returning from interrupt (as demonstrated by DOS's prototype TSR, PRINT.COM). Functions 50h and 51h, Set Active PSP Address and Get Active PSP Address, respectively, provide the mechanism for doing so. The former stuffs the segment address of the PSP to be activated (passed in BX) into the slot where DOS stores the address of

the active PSP. Function 51h does just the opposite, returning in BX the segment address of the active PSP.

Because of an anomaly in the way DOS 2.*x* handled stack switches internally, calling either of these functions from inside an interrupt 28h handler without setting the critical error flag will cause the system to crash. You can work around this by obtaining the address of the critical error flag first and manually incrementing that value before invoking function 50h or 51h, and then resetting the flag once the function has returned. This quirk was corrected in DOS 3.0.

Function 5Dh, subfunction 0Ah is the undocumented counterpart to function 59h (Get Extended Error Information). DOS uses it to set the register values that will be returned by the next call to function 59h. TSRs use it to restore the extended error information obtained with a call to function 59h prior to performing file I/O themselves. By doing so, they preserve any error information that was pending when they were invoked.

DOS'S LIST OF LISTS

Easily the most informative function DOS provides, function 52h returns the address of a list DOS maintains internally that contains a wealth of information, including pointers to other lists (hence the term *list of lists*). From the table whose address is returned by this function (and whose format is dependent on the DOS version), you can ascertain, for example:

- where the first memory control block is located in memory, useful for tracking the chain of memory control blocks DOS sets up;
- where the chain of drive parameter blocks begins (the address of individual drive parameter blocks can be obtained with functions 1Fh and 32h);
- where DOS locates the CON, CLOCK$, and NUL device drivers, and, from the address of the NUL driver, where the chain of device drivers begins (and thus where any other device driver is located);
- where DOS keeps its system file tables, in which information about open files is stored;
- the number of BUFFERS specified in CONFIG.SYS, where these buffers are located, and what's in them;
- the number of logical drives in the

UNDOCUMENTED DOS INTERRUPTS AND FUNCTION CALLS

Interrupt 21h, Function 1Fh: Get Drive Parameter Block for Default Drive

Call with: AH = 1Fh

Returns: AL = Return code
FFh: Error, drive does not exist
Not FFh: No Error, drive exists
DS:BX = Address of drive parameter block

Description: Returns the segment:offset address of the drive parameter block that corresponds to the current drive. The format of the block is shown below.

Drive Parameter Block Format:

	Offset	Size	Description
All Versions of DOS			
	00	Byte	Drive number (0 = A:, 1 = B:, and so on)
	01	Byte	Unit number within device driver
	02	Word	Number of bytes per sector
	04	Byte	Number of sectors per cluster (minus 1)
	05	Byte	Log base 2 of the cluster size
	06	Word	Number of reserved sectors
	08	Byte	Number of file allocation tables
	09	Word	Number of root directory entries
	0B	Word	Sector where data space starts
	0D	Word	Number of the highest cluster
DOS 2.*x*			
	0F	Byte	Number of sectors per file allocation table
	10	Word	Sector where root directory starts
	12	DWord	Address of device header
	16	Byte	Media descriptor byte
	17	Byte	Access indicator byte FFh: Disk has not been accessed 00h: Disk has been accessed
	18	DWord	Address of next parameter block
	1C	Word	Starting cluster of current directory
	1E	64 bytes	Current directory (ASCIIZ string)
DOS 3.*x*			
	0F	Byte	Number of sectors per file allocation table
	10	Word	Sector where root directory starts
	12	DWord	Address of device header
	16	Byte	Media descriptor byte
	17	Byte	Access indicator byte FFh: Disk has not been accessed 00h: Disk has been accessed
	18	DWord	Address of next parameter block
	1C	Word	Starting cluster in search for free space
	1E	Word	Unknown (always FFFFh)
DOS 4.*x*			
	0F	Word	Number of sectors per file allocation table
	11	Word	Sector where root directory starts
	13	DWord	Address of device header
	17	Byte	Media descriptor byte
	18	Byte	Access indicator byte FFh: Disk has not been accessed 00h: Disk has been accessed
	19	DWord	Address of next parameter block
	1D	Word	Starting cluster in search for free space
	1F	Word	Unknown

Interrupt 21h, Function 32h: Get Drive Parameter Block for Specified Drive

Call with: AH = 32h
DL = Drive number (0 = default, 1 = A:, 2 = B:, and so on)

Returns: AL = Return code
FFh: Specified drive does not exist
Not FFh: Specified drive exists
DS:BX = Address of drive parameter block

Description: Returns the segment:offset address of the drive parameter block that corresponds to the specified drive. The format of the block is identical to the one shown for Function 1Fh.

Interrupt 21h, Function 34h: Get Address of InDOS Flag

Call with: AH = 34h

Returns: ES:BX = Address of InDOS flag

CONTINUES

Figure 1: These are some of the most interesting and useful of the undocumented DOS functions. They are marked *reserved* in most DOS technical references.

system and what LASTDRIVE value was specified in CONFIG.SYS.

Ever wonder how programs such as *Manifest* and PC Magazine Labs' own *MicroScope* (included with the *PC Magazine* Benchmark Series, available on PC MagNet) find out what they do about the structure of DOS? Function 52h provides most of the information. And by using this information, it's possible to snoop around inside DOS and draw conclusions that could never be made by inference alone. An important note related to the values returned in the list of lists is that the byte at offset 17h in DOS 2.*x*, 22h in DOS 3.*x* and 4.*x*, doesn't spell out the address of the NUL device driver; it's the actual *beginning* of the NUL device driver.

NUL always starts at offset 17h within this segment in DOS 2.*x* and at offset 22h in DOS 3.*x* and 4.*x*. And since NUL is always the first driver in the chain of device drivers that DOS loads into memory, you can use this as a starting point to traverse the chain of device drivers—both system and installable. Since each device driver header contains a pointer to the next driver in the chain, you don't need to know where the NUL driver ends.

OTHER DOS INTERRUPTS

When a DOS function is called to read from the keyboard or serial port, it doesn't return until the input request is satisfied. The InDOS flag remains set the entire time, effectively blocking out any TSR that doesn't dare execute an interrupt 21h until InDOS is cleared. It's actually safe to use DOS functions higher than 0Ch while lower-numbered DOS functions are processed, because higher-numbered functions use a stack that's separate from the one used by functions numbered 01h through 0Ch.

Interrupt 28h, the *DOS Idle Interrupt*, provides an undocumented means for TSRs to make calls to interrupt 21h while DOS processes function calls that leave InDOS set for extended periods of time. But InDOS is a simple flag that provides no indication of which function is currently being processed. To compensate, when DOS waits for user input from functions 01h, 03h, 07h, 08h, and 0Ah, it executes interrupt 28h over and over to signal TSRs that, although InDOS is set, it's safe to use functions above 0Ch. Hence

UNDOCUMENTED DOS INTERRUPTS AND FUNCTION CALLS

Description: Returns the segment:offset address of the 1-byte InDOS flag, a flag DOS sets to indicate when it is processing an interrupt 21h function call. When InDOS is zero, it's safe for programs to call interrupt 21h. When it's nonzero, a call to interrupt 21h may crash the system.

Interrupt 21h, Function 37h: Get or Set Switch Character

Call with: AH = 37h
AL = Subfunction code
00h: Read switch character
01h: Set switch character
DL = Switch character (for subfunction 01h)

Returns: DL = Switch character (for subfunction 00h)

Description: Retrieves or alters the character DOS recognizes as a switch on the command line (/C, -C, for example)

Interrupt 21h, Function 50h: Set Active PSP Address

Call with: AH = 50h
BX = Segment address of PSP to activate

Returns: Nothing

Description: Activates the indicated PSP. Under DOS 2.*x*, invoking this function from inside an interrupt 28h handler without first setting the critical error flag will crash the system.

Interrupt 21h, Function 51h: Get Active PSP Address

Call with: AH = 51h

Returns: BX = Segment address of active PSP

Description: Returns the segment address of the active PSP. Under DOS 2.*x*, invoking this function from inside an interrupt 28h handler without first setting the critical error flag will crash the system.

Interrupt 21h, Function 52h: Get List of Lists

Call with: AH = 52h

Returns: ES:BX = Address of list

Description: Returns the segment:offset address of an internal list DOS maintains that spells out the address of the first memory control block, the addresses of the CLOCK$, CON, and NUL device drivers, and more. The format of the list, commonly called the "list of lists," is shown below.

List of Lists Format:

	Offset	Size	Description
All Versions of DOS			
	02	Word	Segment address of first memory control block
	00	DWord	Address of first drive parameter block
	04	DWord	Address of list of DOS file tables
	08	DWord	Address of CLOCK$ device driver
	0C	DWord	Address of CON device driver
DOS 2.*x*			
	10	Byte	Number of logical drives
	11	Word	Maximum number of bytes per block, for block devices (usually 512)
	13	DWord	Address of first disk buffer
	17	Byte	Beginning of the NUL device driver
DOS 3.*x* and DOS 4.*x*			
	10	Word	Maximum number of bytes per block, for block devices (usually 512)
	12	DWord	Address of first disk buffer
	16	DWord	Address of drive data array
	1A	DWord	Address of FCB table
	1E	Word	Size of FCB table
	20	Byte	Number of block devices
	21	Byte	LASTDRIVE value (default = 5)
	22	Byte	Beginning of the NUL device driver
	34	Byte	Number of JOINed drives

Interrupt 21h, Function 5Dh, Subfunction 06h: Get Critical Error Flag Address

Call with: AH = 5Dh
AL = 06h (subfunction code)

Returns: DS:SI = Address of the critical error flag

Description: Returns the address of the 1-byte critical error flag. A nonzero value indicates a critical error has occurred in the system. When this flag is set, calls to the DOS kernel should be restricted until processing is complete.

CONTINUES

the name *DOS Idle:* DOS is passing the word that it's sitting in an idle loop waiting for input. All a TSR has to do is tap into interrupt 28h and, if InDOS is set, use the interrupt as a mechanism for gaining control. To detect the occurrence of an interrupt 28h, you have to set up an interrupt handler to trap it. As soon as you've trapped it, you've gained control of the CPU.

Interrupt 29h, the *Fast Console Output Interrupt,* is DOS's undocumented mechanism for passing characters to the console device. To output a character, DOS places the ASCII code in AL and executes an INT 29h. The default console driver, CON, fields the interrupt and uses function 0Eh in the video BIOS (the write- teletype routine) to output the character to the screen. When ANSI.SYS is installed, it takes over interrupt 29h processing from CON and uses interrupt 10h, function 09h to write characters to the screen.

Interrupt 2Eh is also known as the back door to the command processor. Passing a string to DOS with this function causes it to be executed by the top-level command processor, just as if it had been typed at the command prompt. Programmers must be careful to save important registers (including SS and SP) before calling interrupt 2Eh, because it, like DOS's EXEC function, destroys everything but CS and IP. In addition, if the command processor is COMMAND.COM, there must be enough memory available for COMMAND.COM to load the transient portion of itself back into memory, if necessary. This means that if you invoke interrupt 2Eh from a .COM file, you must shrink your own program's memory allocation with a call to function 4Ah first.

Be careful about how you call interrupt 2Eh. If you aren't, you can crash your system in certain very common circumstances. The main one is if the program you're using is running under a batch file. Since INT 2Eh is nonreentrant, DOS uses it to run batch files. So if you run a batch file using INT 2Eh from your program, your system will crash.

CAVEATS

There are two pitfalls to watch out for in using these undocumented functions and interrupts. The first is that they're not supported in all versions of DOS. To the best of my knowledge, all the ones listed here are available in Version 2.0 and later, with the exception of the following:

Interrupt 21h, function 5Dh, subfunction 06h (Get Critical Error Flag Address), which first appeared in DOS 3.0; interrupt 21h, function 5Dh, subfunction 0Ah (Set Extended Error Information), which first appeared in DOS 3.1; interrupt 21h, function 60h (Return Fully Qualified Filename), which first appeared in DOS 3.0.

The second caveat is that none of these functions is guaranteed to be supported in future versions. That's one reason some of them were originally left undocumented: so that Microsoft wouldn't be committed to supporting them in future releases of the operating system. Most, however, have become so widely used that they're likely to be retained in future versions so that subsequent releases of DOS won't be incompatible with today's applications.

ASK THE TUTOR

The Tutor solves practical problems and explains techniques for using your hardware and software more productively. Questions about DOS and systems in general are answered here. To have your questions answered, write to Tutor, *PC Magazine,* One Park Avenue, New York, NY 10016, or upload them to PC MagNet (see page 8 for access instructions). We're sorry, but we are unable to answer questions individually. ■

UNDOCUMENTED DOS INTERRUPTS AND FUNCTION CALLS

Interrupt 21h, Function 5Dh, Subfunction 0Ah: Set Extended Error Information

Call with: AH = 5Dh
AL = 0Ah (subfunction code)
DS:DX = Address of extended error information table

Returns: Nothing

Description: Permits programs to set the extended error information returned by function 59h (Get Extended Error Information). The format of the extended error information table is shown below.

Extended Error Information Table Format:

Offset	Size	Description
00	Word	Value function 59h will return in AX
02	Word	Value function 59h will return in BX
04	Word	Value function 59h will return in CX
06	Word	Value function 59h will return in DX
08	Word	Value function 59h will return in SI
0A	Word	Value function 59h will return in DI
0C	Word	Value function 59h will return in DS
0E	Word	Value function 59h will return in ES
10	Word	Always 0
12	Word	Always 0
14	Word	Always 0

Interrupt 21h, Function 60h: Return Fully Qualified Filename

Call with: AH = 60h
DS:SI = Address of ASCIIZ filename
ES:DI = Address of buffer to receive fully qualified filename

Returns: Nothing

Description: Accepts an ASCIIZ (zero-delimited) filename, which may or may not be fully qualified, and converts it into an unambiguous and fully qualified filename that includes drive and path designators.

Interrupt 28h: DOS Idle

Description: Interrupt 28h, the DOS Idle Interrupt, is repeatedly called while DOS functions 01h, 03h, 07h, 08h, and 0Ah wait for input to permit TSRs to invoke DOS services when the InDOS flag is set.

Interrupt 29h: Fast Console Output

Call with: AL = Character to output to console device

Returns: Nothing

Description: Called by DOS to output characters to the console device (CON or ANSI.SYS).

Interrupt 2Eh: Execute Command

Call with: DS:SI = Address of command string (delimited by the character 0Dh—the carriage return character), with the first byte set equal to the length of the string (the length should include the carriage return at the end)

Returns: Nothing

Description: Passes a command line to the command processor and executes it.

ENDS